Claims

What is claimed is:

1. A method, comprising executing an algorithm by a processor of a computer system, said executing said algorithm comprising sorting S sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of the computer system prior to said sorting, S being at least 2, each sequence of the S sequences comprising contiguous fields of bits, said sorting comprising executing program code at nodes of a linked execution structure, said executing program code being performed in a sequential order with respect to said nodes, said executing program code including:

masking the contiguous fields of the S sequences in accordance with a mask whose content is keyed to the field being masked, said sequential order being a function of an ordering of masking results of said masking; and

outputting each sequence of the S sequences or a pointer thereto to an output array of the memory device whenever said masking places said each sequence in a leaf node of the linked execution structure.

2. The method of claim 1, wherein said sorting does not include comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

3. The method of claim 1, wherein the sorting execution time is a linear function of a sequence length comprised by each sequence of the S sequences.

4. The method of claim 1, wherein the sorting execution time is a linear or less than linear function of S.

5. The method of claim 1, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to said sorting, with respect to said associated values.

6. The method of claim 1, wherein the sorting execution time is a decreasing function of a data density of the S sequences.

7. The method of claim 1, wherein the sorting execution time is a saddle-shaped function of a width W of the mask at a fixed data density of the S sequences.

8. The method of claim 7, wherein W is a constant with respect to said contiguous fields.

9. The method of claim 7, wherein W is variable with respect to said contiguous fields.

10. The method of claim 7, wherein executing said algorithm further comprises selecting W so as

to minimize the sorting execution time at the data density of the S sequences.

11. The method of claim 1, wherein said program code is a modular procedure, and wherein said executing program code further includes recursively calling the modular procedure from within the modular procedure.

12. The method of claim 1, wherein said executing program code further includes counter-controlled looping through the nodes of the linked execution structure.

13. The method of claim 1, wherein the S sequences each represent a variable-length character string.

14. The method of claim 1, wherein the S sequences each represent a fixed-length character string.

15. The method of claim 1, wherein the S sequences consist of S fixed-length words such that each of said words represents an integer.

16. The method of claim 1, wherein the S sequences consist of S fixed-length words such that each of said words represents a floating point number.

17. The method of claim 1, wherein S is at least 1000, and wherein the mask has a width such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via execution of a Quicksort sorting algorithm by said processor.

18. A computer program product, comprising:

a computer usable medium having a computer program embodied therein, said computer readable program comprising an algorithm for sorting S input sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of a computer system prior to said sorting, S being at least 2, each sequence comprising contiguous fields of bits, said algorithm adapted to perform said sorting by executing program code at nodes of a linked execution structure, said executing program code being performed in a sequential order with respect to said nodes, said executing program code including:

masking the contiguous fields of the S sequences in accordance with a mask whose content is keyed to the field being masked, said sequential order being a function of an ordering of masking results of said masking; and

outputting each sequence of the S sequences or a pointer thereto to an output array of the memory device whenever said masking places said each sequence in a leaf node of the linked execution structure.

19. The computer program product of claim 18, wherein said algorithm is not adapted to execute comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

20. The computer program product of claim 18, wherein the sorting execution time is a linear

function of a sequence length comprised by each sequence of the S sequences.

21. The computer program product of claim 18, wherein the sorting execution time is a linear or less than linear function of S.

22. The computer program product of claim 18, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to said sorting, with respect to said associated values.

23. The computer program product of claim 18, wherein the sorting execution time is a decreasing function of a data density of the S sequences.

24. The computer program product of claim 18, wherein the sorting execution time is a saddle-shaped function of a width W of the mask at a fixed data density of the S sequences.

25. The computer program product of claim 24, wherein W is a constant with respect to said contiguous fields.

26. The computer program product of claim 24, wherein W is variable with respect to said contiguous fields.

27. The computer program product of claim 24, wherein W has a value that minimizes the sorting execution time at the data density of the S sequences.

28. The computer program product of claim 18, wherein said program code is a modular procedure, and wherein said executing program code further includes recursively calling the modular procedure from within the modular procedure.

29. The computer program product of claim 18, wherein said executing program code further includes counter-controlled looping through the nodes of the linked execution structure.

30. The computer program product of claim 18, wherein the S sequences each represent a variable-length character string.

31. The computer program product of claim 18, wherein the S sequences each represent a fixed-length character string.

32. The computer program product of claim 18, wherein the S sequences consist of S fixed-length words such that each of said words represents an integer.

33. The computer program product of claim 18, wherein the S sequences consist of S fixed-length words such that each of said words represents a floating point number.

34. The computer program product of claim 18, wherein S is at least 1000, and wherein the mask has a width such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via execution of a Quicksort sorting algorithm by said processor.

35. A method, comprising executing an algorithm by a processor of a computer system, said executing said algorithm comprising sorting S sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of the computer system prior to said sorting, S being at least 2, each sequence of the S sequences comprising K contiguous fields denoted left to right as F_1, F_2, ..., F_K with corresponding field widths of W_1, W_2, ..., W_K, said sorting comprising the steps of:

designating S memory areas of the memory device as A_1, A_2, , ..., A_S;

setting an output index P = 0 and a field index Q = 0;

providing a node E having S elements stored therein, said S elements consisting of the S sequences or S pointers respectively pointing to the S sequences; and

executing program code, including determining a truth or falsity of an assertion that the elements in node E collectively include or point to no more than one unique sequence U of the S sequences, and if said assertion is determined to be false:

then generating C child nodes from node E, each child node including all elements in node E having a unique value of field F_{Q+1}, said child nodes denoted as E_0, E_1, ..., E_{C-1} having associated field F_{Q+1} values of V_0, V_1, ..., V_{C-1}, said child nodes E_0, E_1, ..., E_{C-1} being sequenced such that $V_0<V_1< ...<V_{C-1}$, said generating followed by incrementing Q by 1, said incrementing Q followed by iterating from an index I=0 to I=C-1 in steps of 1, wherein iteration I includes setting $E=E_I$ followed by executing the program code recursively at a next level of recursion for the node E;

else for each element in node E: incrementing P by 1, next storing in A_P either U or the element pointing to U, and lastly if the program code at all of said levels of recursion has not been fully executed then resuming execution of said program code at the most previous level of recursion at which the program code was partially but not fully executed else exiting the algorithm.

36. The method of claim 35, wherein said sorting does not include comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

37. The method of claim 35, wherein the sorting execution time is a linear function of a sequence length comprised by each sequence of the S sequences.

38. The method of claim 35, wherein the sorting execution time is a linear or less than linear function of S.

39. The method of claim 35, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to said sorting, with respect to said associated values.

40. The method of claim 35, wherein the sorting execution time is a decreasing function of a data density of the S sequences.

41. The method of claim 35, wherein the S elements consist of the S pointers, wherein the node E having the S sequences prior to said sorting includes a linked list that comprises the S pointers, and wherein each child node having pointers therein includes a linked list that comprises said pointers therein.

42. The method of claim 35, wherein the S sequences each represent a variable-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

43. The method of claim 35, wherein the S sequences each represent a fixed-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

44. The method of claim 35, wherein the S sequences consist of S fixed-length words such that each of the S words has N binary bits, wherein N is at least 2.

45. The method of claim 44, wherein the S words each represent an integer.

46. The method of claim 44, wherein the method further comprises determining a leftmost

significant bit position of the S words collectively, and wherein the leftmost bit position of field F_1 is the leftmost significant bit position of the S words collectively.

47. The method of claim 44, wherein the S words each represent a floating point number having the following fields contiguously ordered from left to right: a sign field, an exponent field, and a mantissa field.

48. The method of claim 35, wherein generating the C child nodes from node E comprises performing (M AND X) or (X AND M) with a mask M for each sequence X in node E, wherein the mask M is keyed to the field F_{Q+1}, and wherein the bit positions of the mask M relating to the field F_{Q+1} each have a 1 bit, and wherein the remaining bit positions of the mask M each have a 0 bit.

49. The method of claim 35, wherein S is at least 1000, and wherein W_1, W_2, ..., W_K is such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via execution of a Quicksort sorting algorithm by said processor.

50. A computer program product, comprising:

a computer usable medium having a computer readable program code embodied therein, said computer readable program comprising an algorithm for sorting S input sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of a computer system prior to said sorting, S being at least 2, each sequence of the S sequences comprising K contiguous fields denoted left to right as F_1, F_2, ..., F_K with corresponding field widths of W_1, W_2, ..., W_K, said algorithm adapted to perform said sorting by executing the steps of:

designating S memory areas of the memory device as A_1, A_2, , ..., A_S;

setting an output index P = 0 and a field index Q = 0;

providing a node E having S elements stored therein, said S elements consisting of the S sequences or S pointers respectively pointing to the S sequences; and

executing program code, including determining a truth or falsity of an assertion that the elements in node E collectively include or point to no more than one unique sequence U of the S sequences, and if said assertion is determined to be false:

then generating C child nodes from node E, each child node including all elements in node E having a unique value of field F_{Q+1}, said child nodes denoted as E_0, E_1, ..., E_{C-1} having associated field F_{Q+1} values of V_0, V_1, ..., V_{C-1}, said child nodes E_0, E_1, ..., E_{C-1} being sequenced such that $V_0<V_1<...<V_{C-1}$, said generating followed by incrementing Q by 1, said incrementing Q followed by iterating from an index I=0 to I=C-1 in steps of 1,

wherein iteration I includes setting $E=E_I$ followed by executing the program code recursively at a next level of recursion for the node E;

else for each element in node E: incrementing P by 1, next storing in A_P either U or the element pointing to U, and lastly if the program code at all of said levels of recursion has not been fully executed then resuming execution of said program code at the most previous level of recursion at which the program code was partially but not fully executed else exiting the algorithm.

51. The computer program product of claim 50, wherein said algorithm is not adapted to execute comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

52. The computer program product of claim 50, wherein the sorting execution time is a linear function of a sequence length comprised by each sequence of the S sequences.

53. The computer program product of claim 50, wherein the sorting execution time is a linear or less than linear function of S.

54. The computer program product of claim 50, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to said sorting, with respect to said associated values.

55. The computer program product of claim 50, wherein the sorting execution time is a decreasing function of a data density of the S sequences.

56. The computer program product of claim 50, wherein the S elements consist of the S pointers, wherein the node E having the S sequences prior to said sorting includes a linked list that comprises the S pointers, and wherein each child node having pointers therein includes a linked list that comprises said pointers therein.

57. The computer program product of claim 50, wherein the S sequences each represent a variable-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

58. The computer program product of claim 50, wherein the S sequences each represent a fixed-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

59. The computer program product of claim 50, wherein the S sequences consist of S fixed-length words such that each of the S words has N binary bits, wherein N is at least 2.

60. The computer program product of claim 59, wherein the S words each represent an integer.

61. The computer program product of claim 59, wherein the algorithm is further adapted to execute determining a leftmost significant bit position of the S words collectively, and wherein the leftmost bit position of field F_1 is the leftmost significant bit position of the S words collectively.

62. The computer program product of claim 59, wherein the S words each represent a floating point number having the following fields contiguously ordered from left to right: a sign field, an exponent field, and a mantissa field.

63. The computer program product of claim 50, wherein generating the C child nodes from node E comprises performing (M AND X) or (X AND M) with a mask M for each sequence X in node E, wherein the mask M is keyed to the field F_{Q+1}, and wherein the bit positions of the mask M relating to the field F_{Q+1} each have a 1 bit, and wherein the remaining bit positions of the mask M each have a 0 bit.

64. The computer program product of claim 50, wherein S is at least 1000, and wherein W_1, W_2, ..., W_K is such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via execution of a Quicksort sorting algorithm by said processor.

65. A method, comprising executing an algorithm by a processor of a computer system, said executing said algorithm comprising sorting S sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of the computer system prior to said sorting, S being at least 2, each sequence of the S sequences comprising K contiguous fields denoted left to right as F_1, F_2, ..., F_K with corresponding field widths of W_1, W_2, ..., W_K, said sorting comprising the steps of:

designating S memory areas of the memory device as A_1, A_2, , ..., A_S;

setting an output index P = 0 and a field index Q = 0;

providing a node E having S elements stored therein, said S elements consisting of the S sequences or S pointers respectively pointing to the S sequences; and

counter-controlled looping through program code, said looping including iteratively executing said program code within nested loops, said executing said program code including determining a truth or falsity of an assertion that the elements in node E collectively include or point to no more than one unique sequence U of the S sequences, and if said assertion is determined to be false:

then generating C child nodes from node E, each child node including all elements in node E having a unique value of field F_{Q+1}, said child nodes denoted as E_0, E_1, ..., E_{C-1} having associated field F_{Q+1} values of V_0, V_1, ..., V_{C-1}, said child nodes E_0, E_1, ..., E_{C-1} being sequenced such that $V_0<V_1<...<V_{C-1}$; said generating followed by incrementing Q by 1, said incrementing Q followed by iterating from an index I=0 to I=C-1 in steps of 1,

wherein iteration I includes setting $E=E_I$ followed by returning to said counter-controlled looping;

else for each element in node E: incrementing P by 1, next storing in A_P either U or the element pointing to U, and lastly if all iterations of said outermost loop have not been executed then returning to said counter-controlled looping else exiting from said algorithm.

66. The method of claim 65, wherein said sorting does not include comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

67. The method of claim 65, wherein the sorting execution time is a linear function of a sequence length comprised by each sequence of the S sequences.

68. The method of claim 65, wherein the sorting execution time is a linear or less than linear function of S.

69. The method of claim 65, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to said sorting, with respect to said associated values.

70. The method of claim 65, wherein the sorting execution time is a decreasing function of a data

density of the S sequences.

71. The method of claim 65, wherein the S sequences each represent a variable-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

72. The method of claim 65, wherein the S sequences each represent a fixed-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

73. The method of claim 65, wherein the S sequences consist of S fixed-length integers such that each of the S integers has N binary bits, wherein N is at least 2.

74. The method of claim 65, wherein the S sequences consist of S fixed-length floating point numbers, each of said floating point numbers having the following fields contiguously ordered from left to right: a sign field, an exponent field, and a mantissa field.

75. The method of claim 65, wherein S is at least 1000, and wherein W_1, W_2, ..., W_K is such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via

execution of a Quicksort sorting algorithm by said processor.

76. A computer program product, comprising:

a computer usable medium having a computer readable program code embodied therein, said computer readable program comprising an algorithm for sorting S input sequences of binary bits in ascending or descending order of a value associated with each sequence and in a time period denoted as a sorting execution time, said S sequences being stored in a memory device of a computer system prior to said sorting, S being at least 2, each sequence of the S sequences comprising K contiguous fields denoted left to right as F_1, F_2, ..., F_K with corresponding field widths of W_1, W_2, ..., W_K, said algorithm adapted to perform said sorting by executing the steps of:

designating S memory areas of the memory device as A_1, A_2, , ..., A_S;

setting an output index P = 0 and a field index Q = 0;

providing a node E having S elements stored therein, said S elements consisting of the S sequences or S pointers respectively pointing to the S sequences; and

counter-controlled looping through program code, said looping including iteratively executing said program code within nested loops, said executing said program code including determining a truth or falsity of an assertion that the elements in node E collectively include or point to no more than one unique sequence U of the S sequences, and if said assertion is determined to be false:

then generating C child nodes from node E, each child node including all elements in node E having a unique value of field F_{Q+1}, said child nodes denoted as E_0, E_1, ..., E_{C-1} having associated field F_{Q+1} values of V_0, V_1, ..., V_{C-1}, said child nodes E_0, E_1, ..., E_{C-1}

being sequenced such that $V_0<V_1< ...<V_{C-1}$; said generating followed by incrementing Q by 1, said incrementing Q followed by iterating from an index I=0 to I=C-1 in steps of 1, wherein iteration I includes setting $E=E_I$ followed by returning to said counter-controlled looping;

else for each element in node E: incrementing P by 1, next storing in A_P either U or the element pointing to U, and lastly if all iterations of said outermost loop have not been executed then returning to said counter-controlled looping else exiting from said algorithm.

77. The computer program product of claim 76, wherein said algorithm is not adapted to execute comparing a value of a first sequence of the S sequences with a value of a second sequence of the S sequences.

78. The computer program product of claim 76, wherein the sorting execution time is a linear function of a sequence length comprised by each sequence of the S sequences.

79. The computer program product of claim 76, wherein the sorting execution time is a linear or less than linear function of S.

80. The computer program product of claim 76, wherein the sorting execution time is essentially independent of an extent to which the S sequences are ordered in the memory device, prior to

said sorting, with respect to said associated values.

81. The computer program product of claim 76, wherein the sorting execution time is a decreasing function of a data density of the S sequences.

82. The computer program product of claim 76, wherein the S sequences each represent a variable-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

83. The computer program product of claim 76, wherein the S sequences each represent a fixed-length character string, wherein each of the S character strings consists of the K contiguous fields, wherein K is a sequence-dependent variable subject to $W_1=W_2= ... =W_K$ = one byte consisting of a fixed number of binary bits for representing one character.

84. The computer program product of claim 76, wherein the S sequences consist of S fixed-length integers such that each of the S integers has N binary bits, wherein N is at least 2.

85. The computer program product of claim 76, wherein the S sequences consist of S fixed-length floating point numbers, each of said floating point numbers having the following fields contiguously ordered from left to right: a sign field, an exponent field, and a mantissa field.

86. The computer program product of claim 76, wherein S is at least 1000, and wherein W_1, W_2, ..., W_K is such that the sorting execution time is less than a Quicksort execution time for sorting the S sequences via execution of a Quicksort sorting algorithm by said processor.